What Are Information Systems?

Based on Chapter 1 of Bennett, McRobb and Farmer:

Object Oriented Systems Analysis and Design Using UML, (3rd Edition), McGraw Hill, 2005.

In This Lecture You Will Learn:

- How to define an Information System (IS)
- Some examples and types of IS
- How to apply basic concepts of systems theory to IS
- How IS are related to organizations

McGregor On-Line Retail Site

- A typical modern IS with:
  - Online catalogue display and shopping cart
  - Back-office systems store stock details, orders, payment transactions, and more
  - Communications link to credit-card processing centre
  - Robot warehouse control system
  - Delivery scheduling

Elements of an IS

- Every IS has:
  - A human activity that needs information
  - Some stored data
  - An input method for entering data
  - Some process that turns the data into information
  - An output method for representing information

The Role of the Computer

- Computers carry out tasks also done by people and by other technologies
  - *Storage*: signalman's memory / hard disk
  - *Display*: Battle of Britain map / PC screen
  - *Calculation*: mental arithmetic / program
  - *Communication*: telephone line / LAN
- Typical advantages of computers:
  - high speed, low cost, reliability

System Transformation

- All useful systems *transform* their inputs into useful outputs
- For IS, both inputs and outputs are typically information
- This *transformation* is the whole reason for building and operating the system

Transformation Example

- McGregor's *Delivery Scheduling System* may have inputs:
  - Information about orders, available stock, customer addresses, vehicle capacities...
- ...And may have outputs:
  - Which orders to load on each vehicle, what route the vehicle should follow...
- How does this benefit McGregor?

Characteristics of Systems

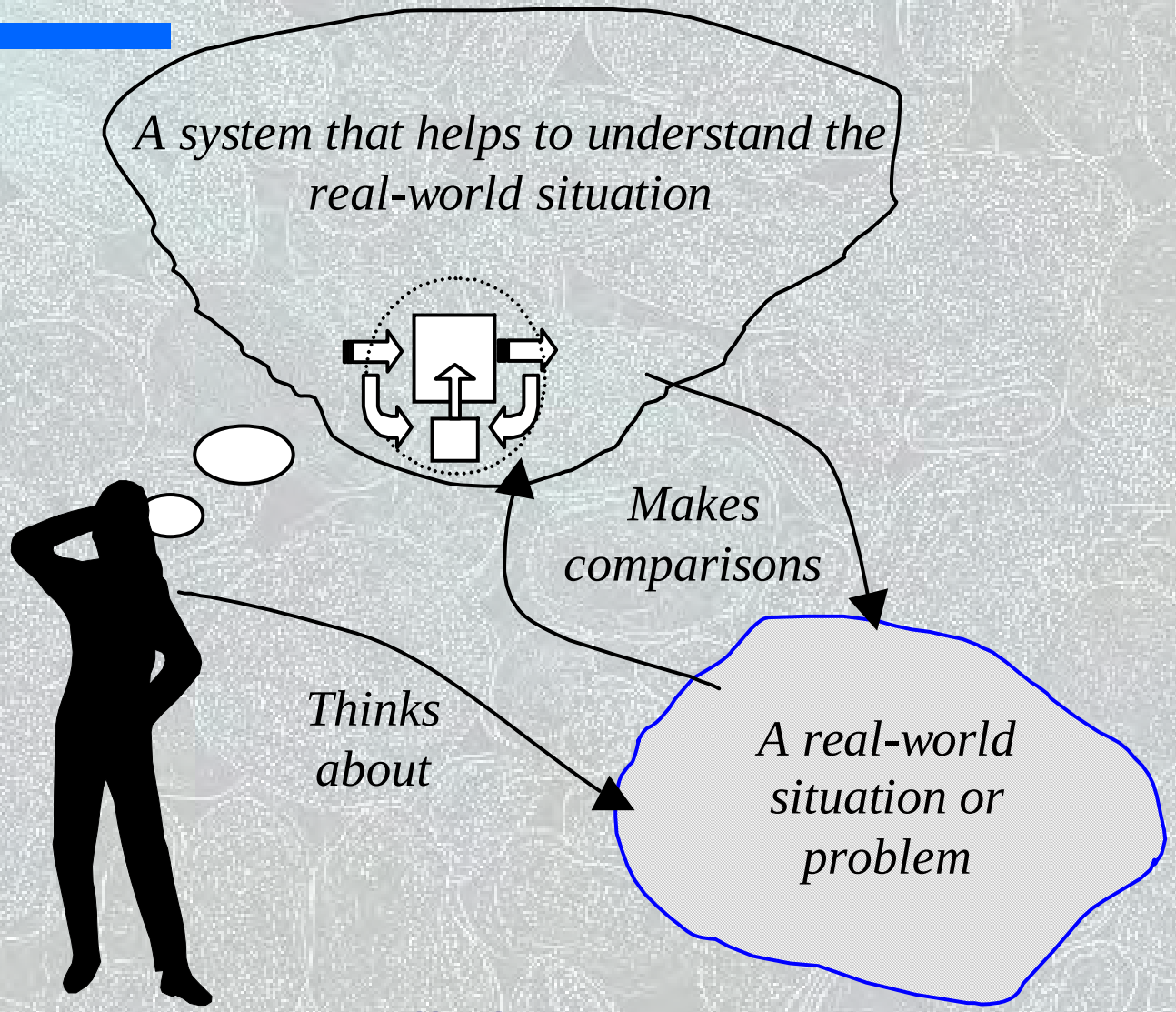
- IS are like any other kind of system
- Every system has:
  - Inputs and outputs
  - A purpose (related to transformation)
  - A boundary and an environment
  - Subsystems and interfaces
  - Control using feedback and feed-forward
  - Some emergent property

Are Systems Real?

Maybe, maybe not!

- Systems thinking is useful because it helps to analyse and understand problems
- What matters is the understanding you achieve
- You can choose to see *anything* as a system, whether or not it really is one

Systems and the Real World



Types of IS

- Information Systems are used to support people's activities
  - Store and retrieve information
  - Carry out calculations
  - Aid communication
  - Control and schedule work
  - Other support ... ?

Types of IS (cont'd)

- Operational Systems assist or control business operations
  - An Accounting System replaces costly and error-prone human clerks
- Management Support Systems help managers to decide or to communicate
  - A Delivery Scheduling System helps decide how to load and route the delivery trucks

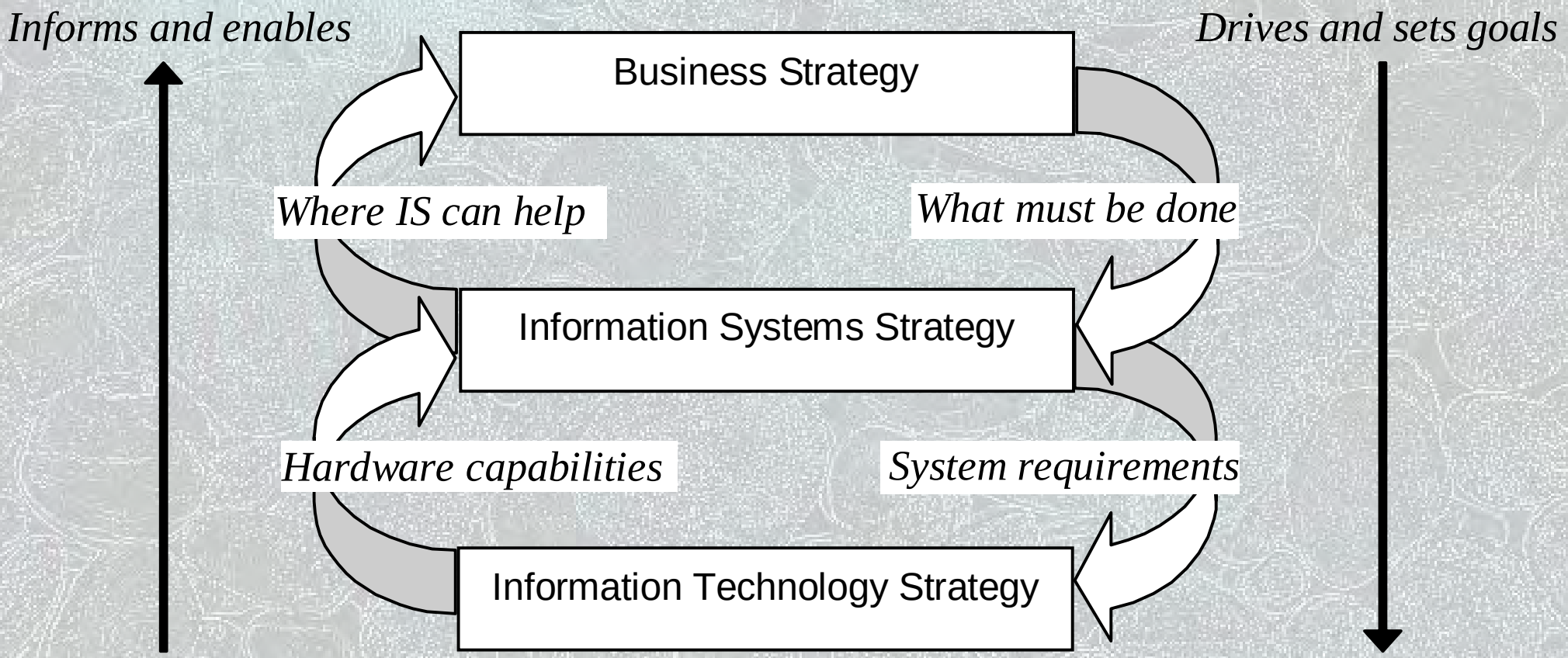
Types of IS (cont'd)

- Real-time Control Systems typically operate physical equipment, often in safety-critical settings
  - Some cars have an Engine Management System to control fuel supply and ignition

How Do IS Relate to the Human Activity System?

- We can view an organization as a system, perhaps with many subsystems
- Ideally, each subsystem helps the overall system fulfil its purpose
- IS are also subsystems and should help to meet goals of people in the organization

Strategy and Planning for IS



Summary

In this lecture you have learned about:

- What an IS is
- Some examples and types of IS
- Some basic concepts of systems theory and how to apply them to IS
- How IS are related to organizations

References

- Bennett, McRobb and Farmer (2005)
- Checkland and Scholes (1990)

(For full bibliographic details, see Bennett, McRobb and Farmer)

Problems in Information Systems Development

Based on Chapter 2 of Bennett, McRobb and Farmer:

Object Oriented Systems Analysis and Design Using UML, (3rd Edition), McGraw Hill, 2005.

In This Lecture You Will Learn:

- The main players in an IS project
- The problems in IS development
- The underlying causes of these problems
- How the *stakeholder* concept helps identify ethical issues in IS development
- The costs of problems and ethical issues

The Main Players

- Three main types of player are involved in an IS development project:
  - Those who will benefit from the system's outputs, directly or indirectly (end-users)
  - Those who commission the project, pay for it or have the power to halt it (owners or sponsors)
  - Those who will produce the software (developers)

What Do We Mean by Problem?

- An IS project may fail before delivery
  - The LSE Taurus project was cancelled
- An IS may fail after delivery
  - The LAS system was withdrawn after implementation
- An IS may be continue to be used, despite causing problems to its users, its owners or its developers

End-user View

- End-users may directly operate the software, or may be more remote, e.g. a manager who receives printed reports
- Typical concerns include:
  - A system that is promised but not delivered
  - A system that is difficult to use
  - A system that doesn't meet its users' needs

Owner View

- Owners care about meeting business needs and about value for money
- Typical concerns include:
  - Projects that overspend their budget (may no longer have a net benefit)
  - Systems that are delivered too late
  - Badly managed projects
  - Systems that are rendered irrelevant by events

Developer View

- IS developers sometimes have a difficult time
  - Budget and time constraints often conflict with doing the job properly
  - Users and owners may not know how to ask for what they really want
  - Technologies, development approaches and business needs all constantly change

Why Things Go Wrong

- Whether a system is delivered or not, many things can go wrong
- Flynn (1998) categorizes the main causes as:
  - Quality problems
  - Productivity problems

Quality Problems

- The wrong problem is addressed
  - Failure to align the project with business strategy
- Wider influences are neglected
  - Project team or business managers don't take account of the system environment
- Incorrect analysis of requirements
  - Poor skills or not enough time allowed
- Project undertaken for wrong reason
  - Technology pull or political push

Productivity Problems

- Users change their minds
- External events
  - E.g. introduction of the Euro currency
- Implementation not feasible
  - May not be known at start of the project
- Poor project control
  - Inexperienced management or political difficulties

Ethics Issues and Stakeholder Problems

- Some IS may affect people far beyond obvious users and owners of the system
  - Cellphone companies collect data about subscribers' calls and physical movements
  - This data can be passed to police and many other government agencies
  - Do you know what data is stored about you? Who by? And what it is used for?

Stakeholder Analysis

- This approach tries to identify everyone affected by a proposed IS
  - Who are the stakeholders?
  - How does the system affect each group?
  - What are their legitimate concerns?
  - Are there any legal implications, e.g. Data Protection Act in the UK?

Summary

In this lecture you have learned about:

- The main players in an IS project, and how they perceive the potential problems
- The origins of the main types of problem
- How stakeholder analysis can help identify ethical impacts of an IS

References

- Flynn (1998)

(For full bibliographic details, see Bennett, McRobb and Farmer)

See also www.ccsr.cse.dmu.ac.uk

Avoiding the Problems

Based on Chapter 3 of Bennett, McRobb and Farmer:

Object Oriented Systems Analysis and Design Using UML, (3rd Edition), McGraw Hill, 2005.

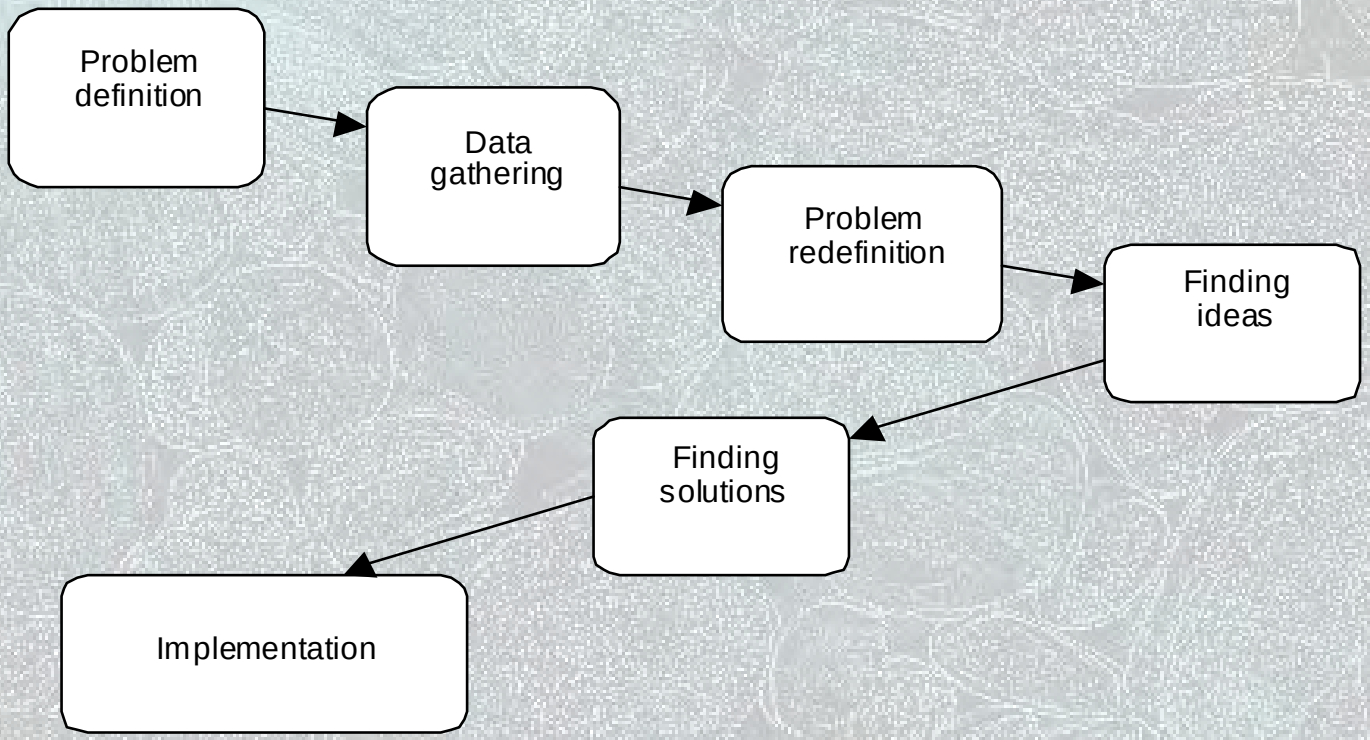
In This Lecture You Will Learn:

- the stages in the waterfall life cycle;
- about prototyping and incremental life cycles;
- the importance of project management;
- how users may be involved in a project;
- the role of CASE tools in systems development.

Problem Solving Model

- Main phases are
  - Data gathering
  - Problem redefinition
    - These focus on understanding what the problem is about
  - Finding ideas
    - Concerned with understanding more about the nature of the problem and possible solutions
  - Finding solutions
  - Implementation

Problem Solving Model



General problem solving model (adapted from Hicks, 1991).

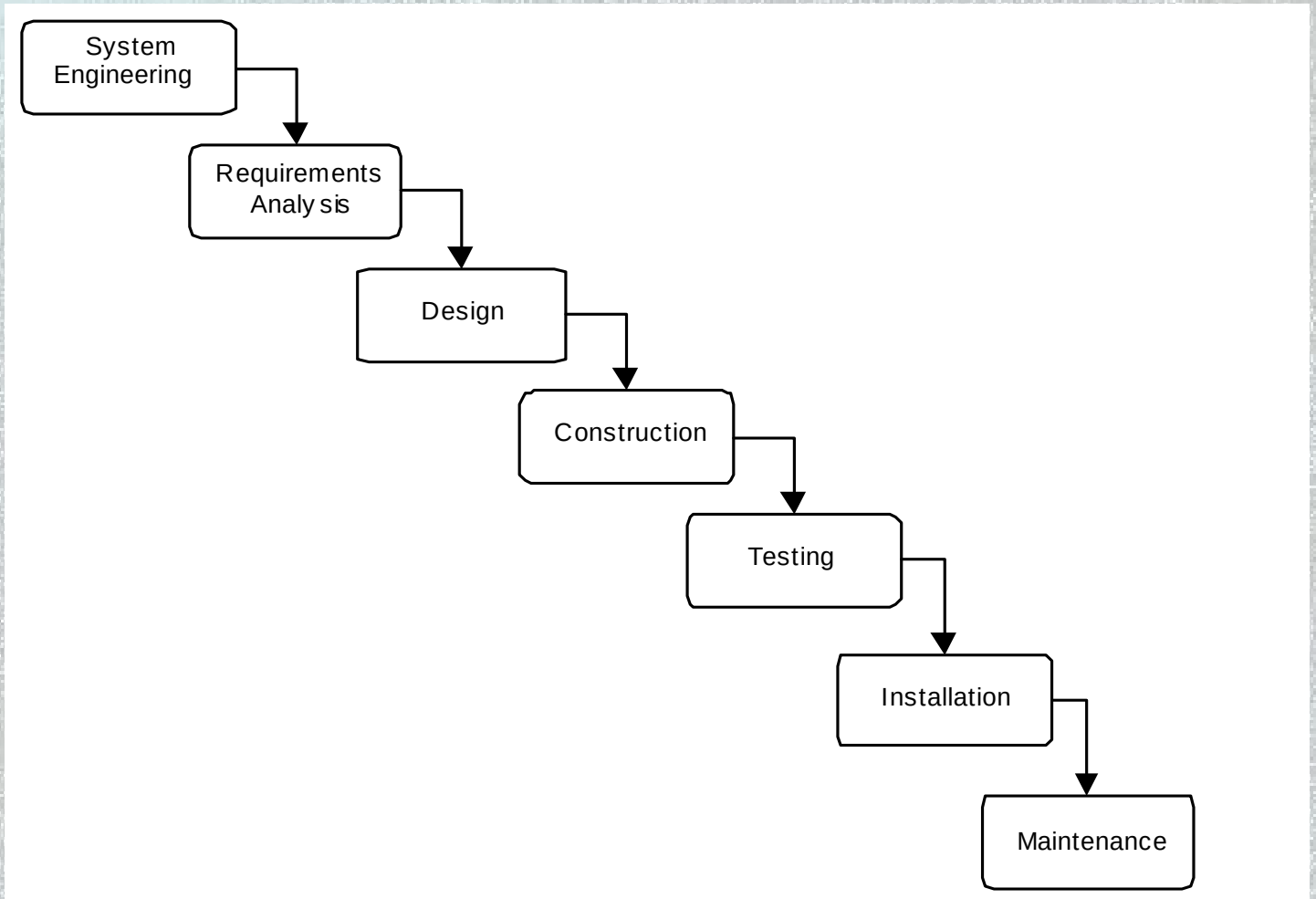
Project Life Cycles

- A distinction should be made between
  - Systems development, which incorporates human, software and hardware elements
  - Software development, which is primarily concerned with software systems
- Two important phases are
  - Strategic Information Systems Planning
  - Business Modelling

Waterfall Life Cycle

- The traditional life cycle (TLC) for information systems development is also known as the waterfall life cycle model.
  - So called because of the difficulty of returning to an earlier phase.
- The model shown here is one of several more or less equivalent alternatives.
  - Typical deliverables are shown for each phase.

Traditional Life Cycle



TLC Deliverables

- Systems Engineering
  - High level architectural specification
- Requirements Analysis
  - Requirements specification
  - Functional specification
  - Acceptance test specifications

Life cycle deliverables (adapted from Sommerville, 1992).

TLC Deliverables

- Design
  - Software architecture specification
  - System test specification
  - Design specification
  - Sub-system test specification
  - Unit test specification

Life cycle deliverables (adapted from Sommerville, 1992).

TLC Deliverables

- Construction
  - Program code
- Testing
  - Unit test report
  - Sub-system test report
  - System test report
  - Acceptance test report
  - Completed system

Life cycle deliverables (adapted from Sommerville, 1992).

TLC Deliverables

- Installation
  - Installed system
- Maintenance
  - Change requests
  - Change request report

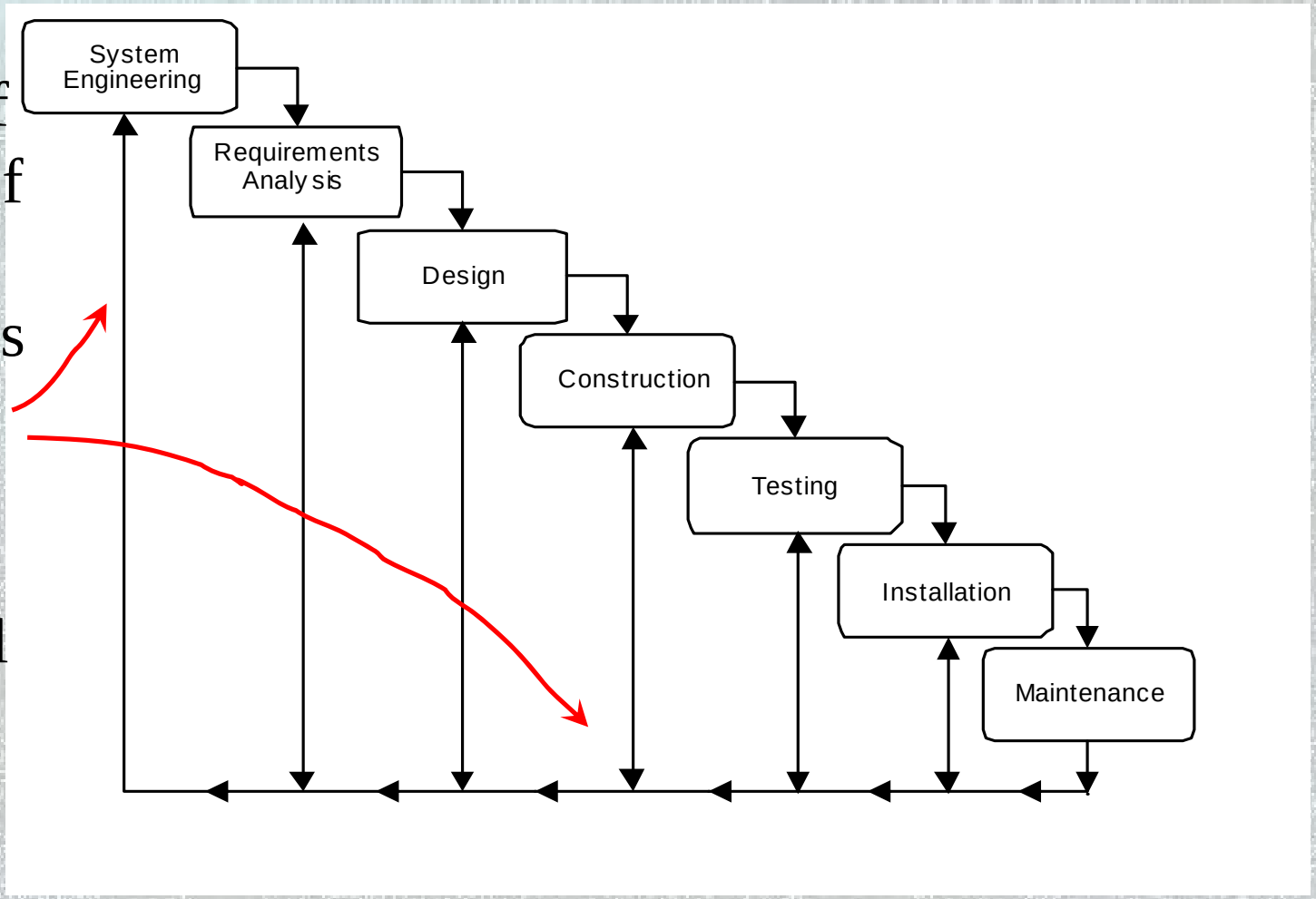
Life cycle deliverables (adapted from Sommerville, 1992).

Problems with TLC

- Real projects rarely follow such a simple sequential life cycle
- Lapsed time between systems engineering and the final installation is long
- Iterations are almost inevitable in real projects but are expensive & problematic with the TLC
- Unresponsive to changes during project as iteration is difficult

TLC with Iteration

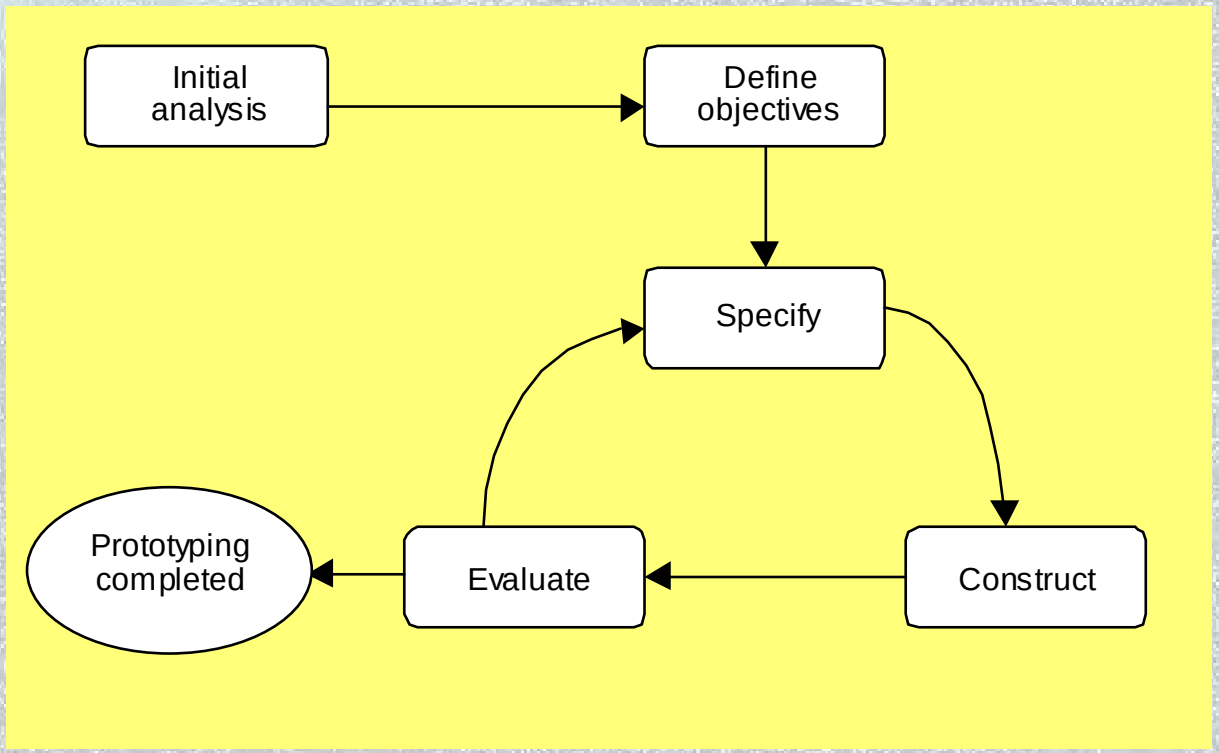
The cost of this form of iteration increases as the project progresses making it impractical and **not** effective



Strengths of TLC

- Tasks in phases may be assigned to specialized teams.
- Project progress evaluated at the end of each phase.
- Can be used to manage projects with high levels of risks.

Prototyping Life Cycle



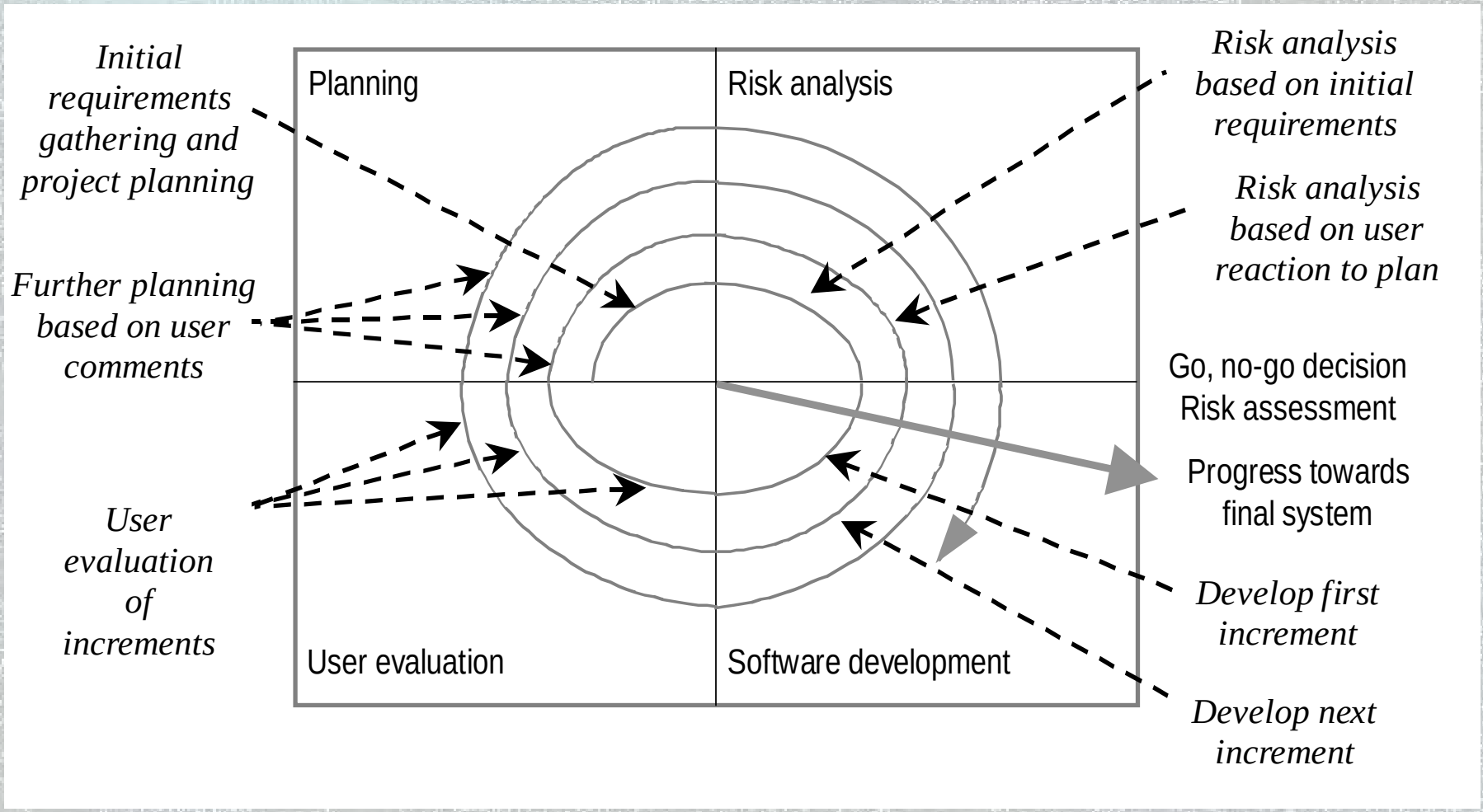
Prototyping – Advantages:

- Early demonstrations of system functionality help identify any misunderstandings between developer and client
- Client requirements that have been missed are identified
- Difficulties in the interface can be identified
- The feasibility and usefulness of the system can be tested, even though, by its very nature, the prototype is incomplete

Prototyping – Problems:

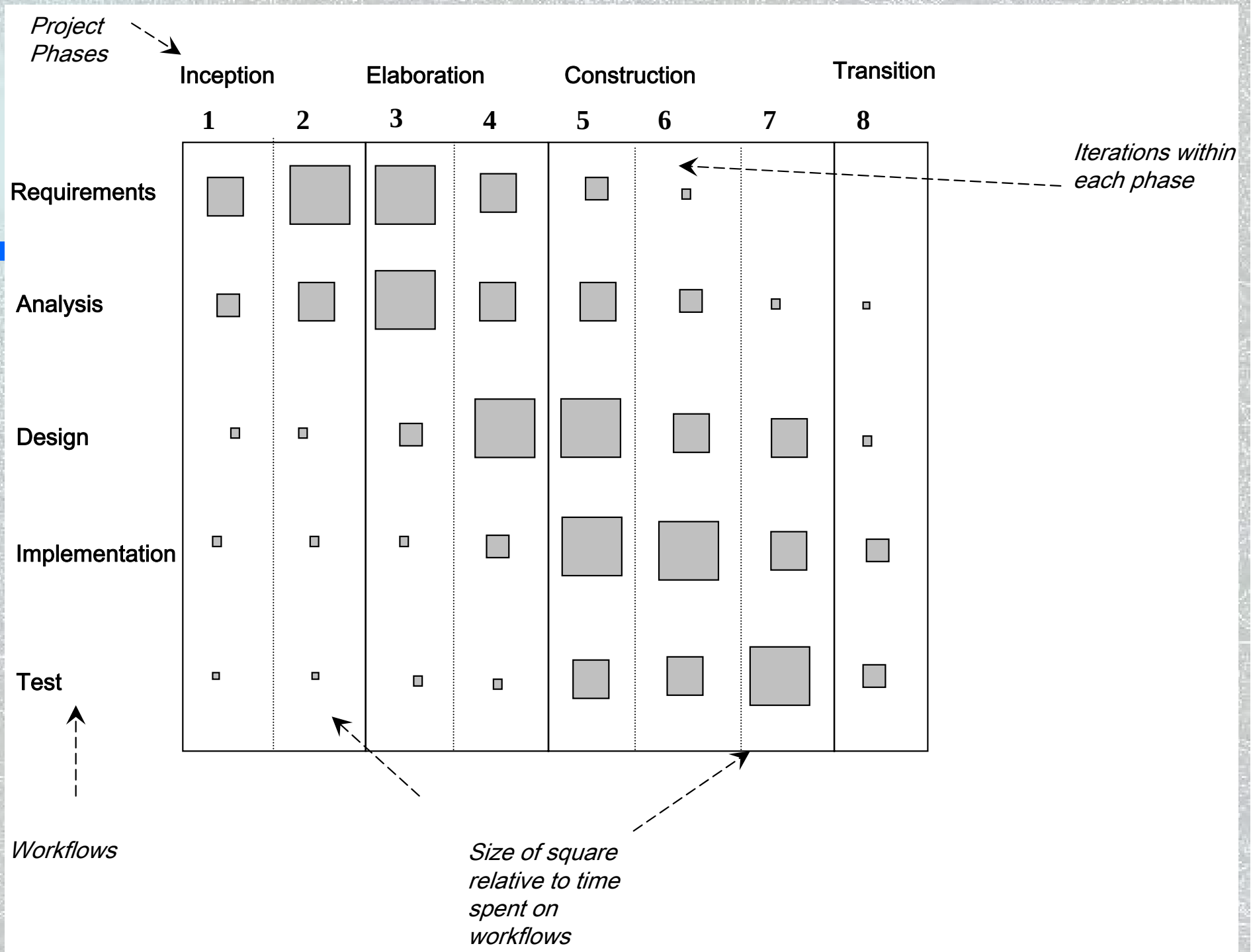
- The client may perceive the prototype as part of the final system
- The prototype may divert attention from functional to solely interface issues
- Prototyping requires significant user involvement
- Managing the prototyping life cycle requires careful decision making

Spiral Model & Incremental Development



Unified Software Development Process

- Captures many elements of best practice
- The phases are:
  - *Inception* is concerned with determining the scope and purpose of the project;
  - *Elaboration* focuses requirements capture and determining the structure of the system;
  - *Construction's* main aim is to build the software system;
  - *Transition* deals with product installation and rollout.


Project Phases
Inception
Elaboration
Construction
Transition
1
2
3
4
5
6
7
8
Iterations within each phase
Requirements
Analysis
Design
Implementation
Test
Workflows
Size of square relative to time spent on workflows

User Involvement

- User's can be involved at various levels
  - As part of the development team (DSDM)
  - Via a consultative approach
  - In fact gathering

Agile Approaches

- Iterative lightweight approach
- Accepts that user requirements will change during development
- XP and DSDM are considered agile
- Non-agile approaches can be viewed as plan-based

Agile Approaches

Manifesto for Agile Software Development

We are uncovering better ways of developing software
by doing and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

The Manifesto for Agile Software Development

Computer Aided Software Engineering

- CASE tools typically provide a range of features including:
  - checks for syntactic correctness;
  - repository support;
  - checks for consistency and completeness;
  - navigation to linked diagrams;

Computer Aided Software Engineering

- Features of CASE tools continued
  - layering;
  - traceability;
  - report generation;
  - system simulation;
  - performance analysis;
  - code generation.

Summary

In this lecture you have learned about:

- the stages in the waterfall life cycle;
- about prototyping and incremental life cycles;
- the importance of project management;
- how users may be involved in a project;
- the role of CASE tools in systems development.

References

- Hicks (1991)
- Sommerville (1992, 2004) and Pressman (2004)
- Jacobson, Booch and Rumbaugh (1999)
- Chapters 5 and 21 of Bennett, McRobb and Farmer include more detail about the Unified Process

(For full bibliographic details, see Bennett, McRobb and Farmer)